

# CS0 : The Beauty, Joy and Awe of Computing

We propose to develop Computer Science 0 (CS0): a new introductory general service course, available to students across the university, to share the beauty, joy and awe of computing (Mcgettrick et al., 2008; Garcia et al., 2009). This course has the potential to serve as a model for a new CollegeBoard Advanced Placement course in the works (Astrachan et al., 2009), which could have national impact. Students will be gently introduced to programming and computational thinking using a new graphical programming language called Scratch (Maloney et al., 2004), with the emphasis on problems relevance to themselves and society. In addition to allowing more opportunities for creativity in the first computer science course, the language has been designed to make learning to program easier by preventing a common frustration for novices, syntax errors. It also supports a computer science "big idea", which is software reuse – it allows students to upload their finished graphical programs to the web which can then be run online in a web browser, downloaded, modified (or, "re-mixed") and re-uploaded. These "Web 2.0" features are the first to be integrated seamlessly into a programming environment, and we are encouraged by the existing active community of worldwide student developers. Finally, the new course will provide the opportunity to broaden participation in computing, a critical component to addressing the current computing enrollment crisis.

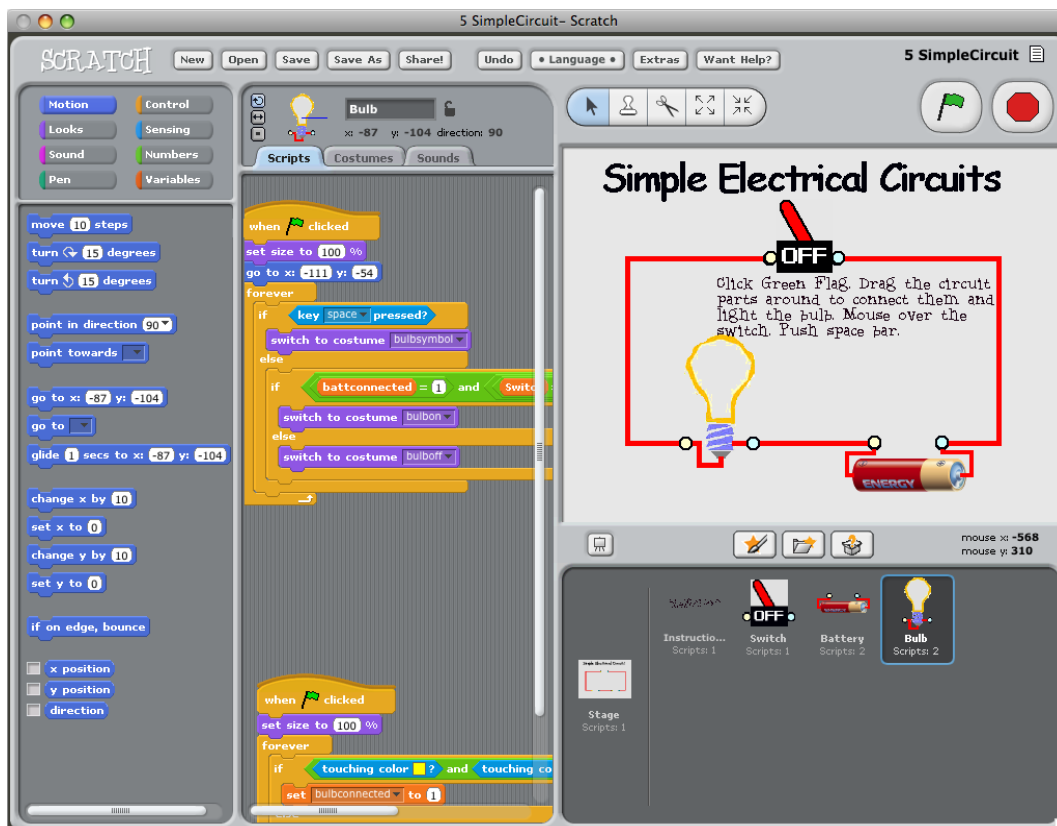


Figure 1: Screenshot of the Scratch interactive development environment running a simple electrical circuit simulation

## Academic Preparation for Computer Science Related Majors

The computer science “61-series”, CS61a, CS61b, and CS61c, forms the foundation for the both the College of Engineering Electrical Engineering and Computer Science major (EECS) and College of Letters and Science Computer Science major (CS). In addition, students in the Cognitive Science major (CogSci) are required to take the first of these courses, *CS61a – Structure and Interpretation of Computer Programs*. This course CS61a requires a level of proficiency with computer programming, as might be gained from the opportunity to enroll in Advanced Placement (AP) computer science in high school.

We have a longstanding goal to provide alternative paths to prepare students for CS61a. The traditional path to CS61a, of taking the AP computer science test, suffers from little participation by populations that are typically under-represented in computer science. In 2007, only 17% of AP computer science test-takers were female. In the same year, black, Hispanic and Latino students comprised only 10.2% of all AP computer science test-takers (College Board, 2008). To broaden access to the EECS, CS and CogSci majors, we need to provide alternatives to AP computer science as preparation for CS61a.

To support students who lacked sufficient preparation to enroll in CS61a, the course CS3 – *Introduction to Symbolic Programming* was developed. This course, based upon the book *Simply Scheme: Introducing Computer Science*, by Brian Harvey (Faculty Advisor) and Matthew Wright has supported students by providing the requisite background in computer science and programming to be successful in CS61a. However, despite the preparation of CS3, students who enter CS61a after taking CS3 have a lower mean grade in CS61a than those who do not take CS3. On average, students who have taken CS3 have a mean score of 0.22 (out of 4.0) lower than students who did not take CS3 (Office of Planning & Analysis, UC Berkeley, 2008).

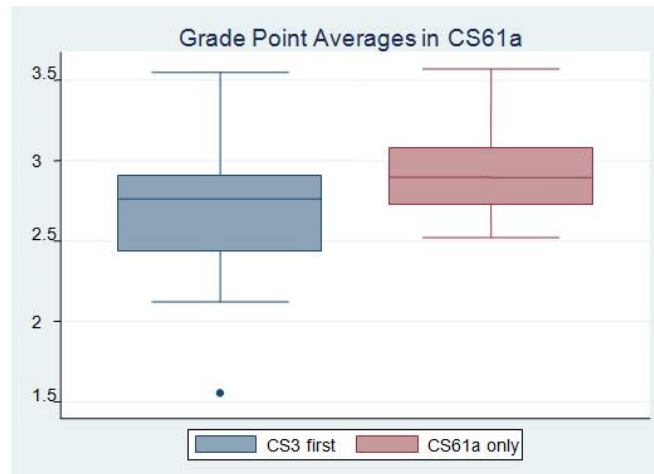
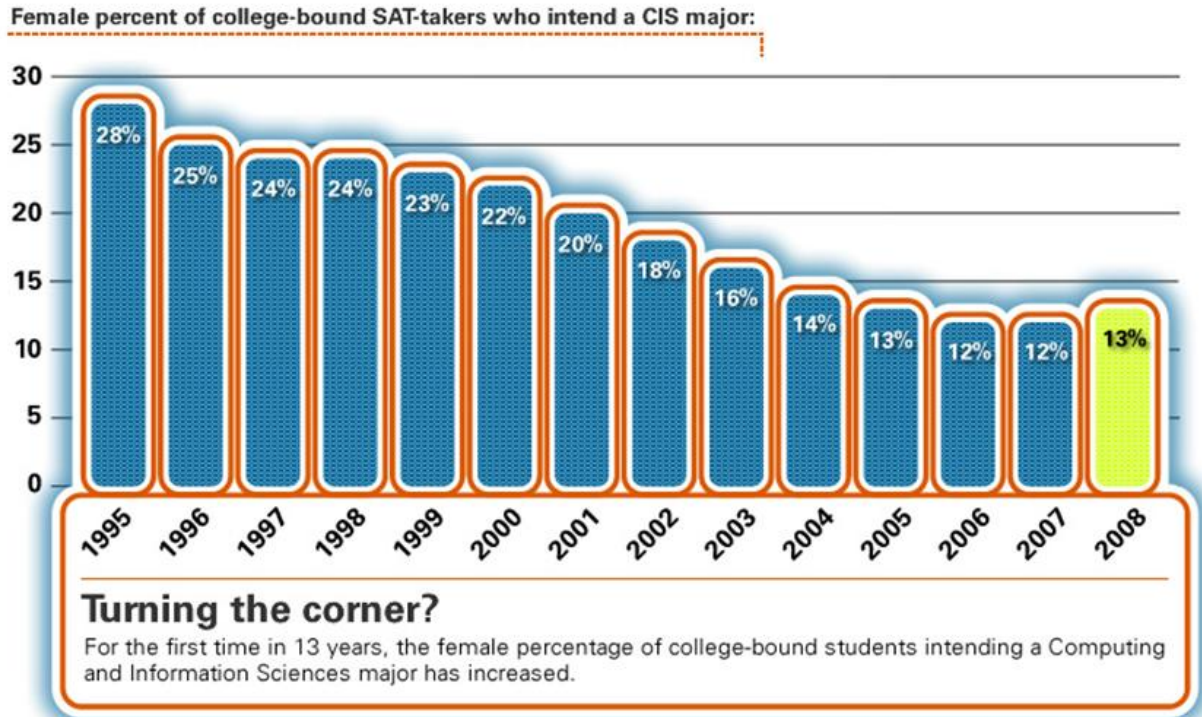


Figure 2: Comparison of grade point average in CS61a for students that did and did not take CS3 before taking CS61a.

There may be additional differences between the students that did and did not take CS3 before taking CS61a. However, one of the design goals of CS0 is to provide a course that puts students with no experience on equal footing with their peers that have computing experience from high school / community college and ensure that the no-experience students can be as successful in the computer science major. To this end, we are investigating additional avenues for supporting students in entering CS61a.

## Broadening Participation

Nationally, there has been attention to the underrepresentation of women in computer science. Although 2008 experienced the first increase in the percentage of female college-bound SAT-takers who intend to major in a Computer and Information Sciences major, the percentage of female students has decreased from a high in 1995 of 28% to only 13% in 2008. (College Board)



Source: College Board

Figure 3: National data of students' reported interest in Computing and Information Science majors.

While female representation at Berkeley in the graduate program and undergraduate majors of EECS and CS is even lower than the national average of college-bound SAT-takers who intend to pursue a related major, Berkeley has experienced a corresponding decrease in female representation (Judson, 2008).

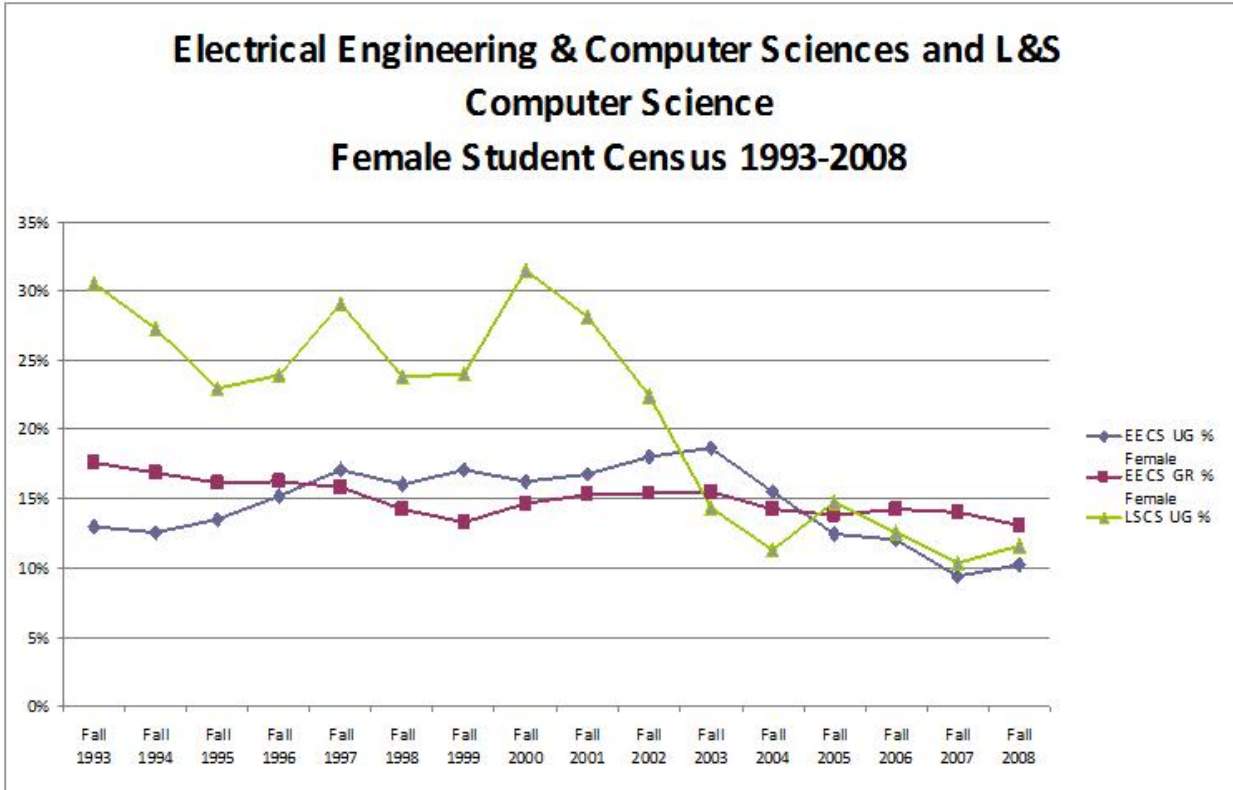


Figure 4: Representation of female students at UC Berkeley in the undergraduate EECS major (EECS UG), the undergraduate CS major (LS CS UG), and the graduate EECS program (EECS GR).

Enrollment in CS3 echoes the national trend of decreased attraction of female students. Enrollment of female students in CS3 for a letter grade has decreased from Fall 1997 to Spring 2008, significant at the 1% level ( $p=0.008$ ). Each course offering accounts for approximately a 0.35% decrease in representation of female students.

However, the participation of female students in CS3 has been far higher than the College Board data would predict. Between the fall of 1997 and the fall of 2008, CS3 averaged 31.4% female enrollment. The representation within the EECS and CS majors of less than 15% appears to highlight a missed opportunity in terms of attracting the more than 30% of female students in CS3 to the major. This pattern suggests we need to find alternative ways to retain these female students that show an initial interest in computer science through enrolling in CS3.

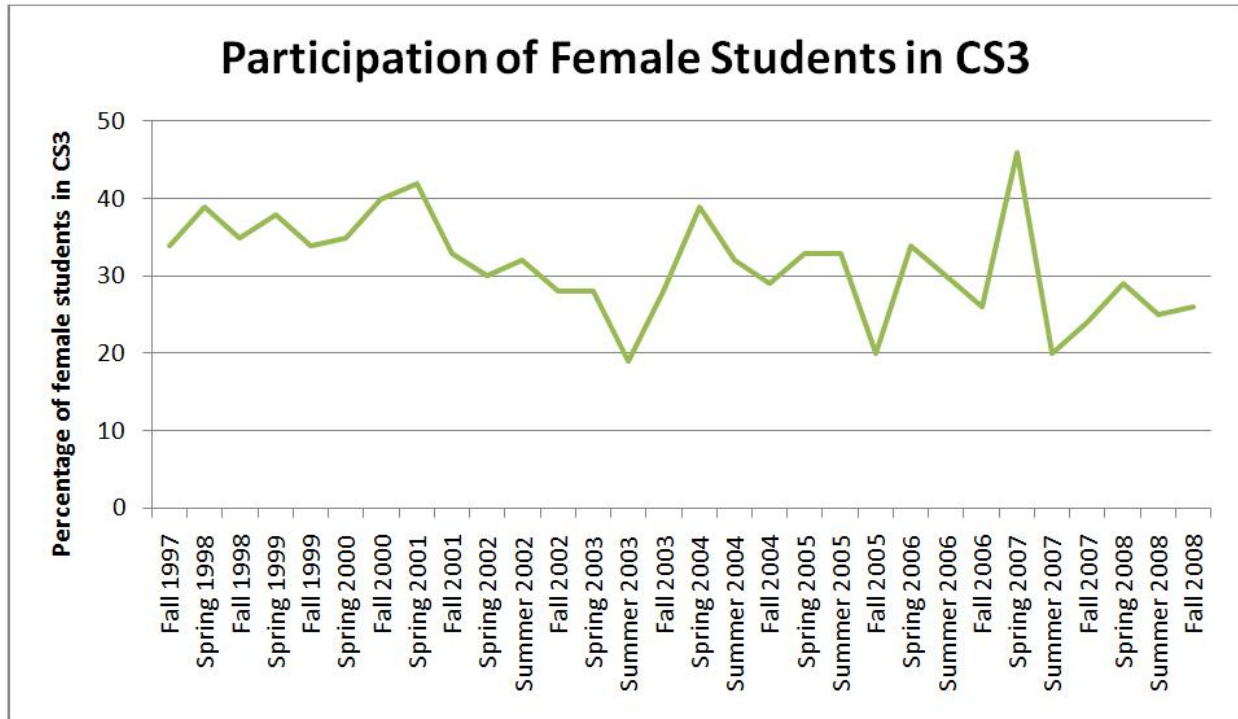


Figure 5: Percentage of female students that took CS3 for a grade.

Research shows that “when [teen girls] hear about computer programming, they conjure up pictures of isolated cubicles, long lines of code, geeky guys, and caffeinated nights. Technology jobs are viewed as working with *things* rather than with *people*.” (Farmer, 2008). Through innovative CS0 curriculum design, we hope to address these stereotypes, and provide assignments that are engaging, and relevant to our students and society.

### **Programming Literacy for Non Computer Science Majors**

Our society presents an increasing need for individuals to have technological competence, including experience writing computer programs (diSessa, 2000). CS0 provides the opportunity to engage students that do not plan to major in computer science, which is our primary target audience. This would allow these students to develop their understanding of and vocabulary for computer science and programming to use in future collaboration.

### **Proposed Course**

Scratch (Maloney et al., 2004), a new programming language developed at Massachusetts Institute of Technology (MIT), was designed to lower the threshold for students to learn to program. However, the language is powerful enough that it can support the curricular needs of a college level introduction to computer science course. Although the computer science topics covered will stay relatively consistent with the original version of CS3, this redesign and new programming language provides the opportunity to change the nature of student assignments. Student projects can be customized with graphics and animation to make them personally meaningful. In addition, student projects can be posted online to be viewed in the Scratch forum, which supports the sharing and collaboration of Scratch developers. For example, instead of writing a text-only program to convert from Fahrenheit to Celsius, students will be creating animations, games and simulations that they can share online with their family and friends.

Although Scratch was originally designed for young students, Harvard University now uses Scratch in their introductory programming course and has found it particularly useful as a way to bridge students, with little programming experience, to programming in Java. (Malan and Leitner, 2007). However, Harvard University uses Scratch only as an introduction in the beginning of their course. Our move to introduce an entirely Scratch course represents a bold move to reshape the face of introductory computer science. The important "big ideas" of computing (e.g., program design, patterns, functional decomposition, concurrency, testing, debugging, software re-use, and recursion) from CS3 will still be addressed.

### **Time-line/Implementation**

In the fall of 2009, we will introduce an experimental section of the course as a freshman/sophomore seminar. With an expected enrollment of 15 students, the course will draw from students across the university. During this first semester, it will be offered on a pass/not pass basis allowing for the evaluation and refinement of the curriculum. In the spring of 2010, we will introduce the course as a concurrent alternative to CS3. This will bring the course to a larger audience in hopes of eventually transitioning the current CS3 course to the new course curriculum and content. In the fall of 2010 and spring of 2011 we will focus on building the strength of the curriculum, while disseminating our results at computer science education conferences, which will allow this curriculum innovation to reach students across the country.

### **Metrics for Success**

In addition to assessing student learning within the new curriculum, we will measure and analyze how students in the course change in their beliefs about, confidence in, and interest in computer science. Longitudinal metrics of success for the new curriculum will be the performance of course participants in CS61a and persistence in computer science courses after the introductory course.

### **Allocation of Funds**

The financial support will support the graduate students and research scientists who will develop the curriculum. In addition to the development of course lectures and assignments, the use of the Scratch programming environment decreases the utility of available infrastructure for grading. Additional assessment criterion and infrastructure will be developed to support these new and innovative assignments. In order to assist the dissemination of results within the community of computer science educators, money will be allocated for travel to conferences where we will present our results from the curriculum development, implementation, and evaluation.

### **Introduction to the Team**

Our team consists of one graduate student, three faculty advisors and two research scientists. There is a close connection between this team and the UC-WISE research group, led by Michael Clancy, which meets weekly to discuss pedagogical issues relevant to education research in the lower division computer science curriculum, and we expect continued collaboration between our two groups.

#### **Colleen Lewis – Graduate Student** (Primary Contact)

ColleenL@berkeley.edu  
5613 Tolman Hall, UC Berkeley  
510-388-7215

Colleen Lewis is a graduate student in Berkeley's Graduate Group in Science and Mathematics Education (SESAME). In this interdisciplinary program she is pursuing her

Masters in computer science (expected December 2009) as well as her PhD in education. Her research focuses on how programming environments such as Scratch support students conceptual development in computer science. She has conducted research in both in- and out-of-classroom contexts and presented her research at the Scratch Conference at Massachusetts Institute of Technology in July 2008. She has designed curriculum and co-taught a summer enrichment program in Scratch for junior-high school students. This summer she will be teaching an additional 60 hours of curriculum in Scratch. She has taught CS3, as both the instructor and as a Graduate Student Instructor, where she won the Outstanding Graduate Student Instructor Award.

**Michael Clancy – Faculty Advisor**

clancy@cs.berkeley.edu  
779 Soda Hall, UC Berkeley

Michael Clancy is a Senior Lecturer and the coordinator for the lower division computer science courses in the EECS Department at University of California, Berkeley. Clancy is one of the leading CS education researchers in the U.S., and his experience spans a wide variety of pedagogy: self-paced instruction, labs, case studies, pair programming, peer instruction, and lab-centric courses. Clancy has long been a proponent of the case study approach to computer science education, and with Marcia Linn has authored two textbooks of case studies and several related papers; He has also run numerous workshops on ways to incorporate this approach into instruction. He has been involved with the Advanced Placement Computer Science Program off and on since 1985 as committee member (committee chair 1987-1992), consultant, and author of two case studies on which the AP CS exam is partly based, most recently "The AP CS Marine Biology Case Study", tested on the 2001-2003 AP CS examinations.

Clancy has participated in the yearly ACM Special Interest Group Symposium on Computer Science Education (SIGCSE), either as paper presenter, panelist, or workshop or tutorial presenter, since 1990, and was a member of the 1999 conference program committee. In 2009, Clancy received the ACM SIGCSE Award for Lifetime Service to the Computer Science Education Community. He serves on the editorial board of Computer Science Education, a primary forum for CS education research. He has coauthored papers presented at the biennial Empirical Studies of Programmers (ESP) conference and has served on the program committees of ESP and all the International Computing Education Research Workshops. He is the director of the UC-WISE project, exploring the use of computing technology to provide hands-on lab-based instruction in the lower-division programming courses.

**Dan Garcia – Faculty Advisor**

ddgarcia@cs.berkeley.edu  
777 Soda Hall, UC Berkeley

Dan Garcia is a Lecturer SOE in the Computer Science Division of the EECS Department at the University of California, Berkeley, and joined the Cal faculty in the fall of 2000. He has won the departmental Diane S. McEntyre Award for Excellence in Teaching in 2002, the departmental Information Technology Faculty Award for Excellence in Undergraduate Teaching in 2004, and was chosen as a UC Berkeley "Unsung Hero" in 2005. He recently earned the highest teaching effectiveness ratings in the history of the department's lower-division introductory courses (tied with one other at 6.7 / 7). He has taught (or co-taught as a graduate student instructor, where he won both departmental and campus outstanding GSI awards) courses in teaching techniques, computer graphics, virtual reality, computer animation, self-paced programming as well as the lower-division introductory curriculum. He is active in SIGCSE, and serves on the ACM Education Board as well as BFOIT, a wonderful Berkeley outreach effort.

He is currently mentoring over seventy undergraduates spread across four groups he founded in 2001 centered around his research, art and development interests in computer graphics, Macintosh OS X programming, computational game theory and computer science education. He also recently co-developed a computing course for all freshman engineers. Dan received his PhD and MS in Computer Science from UC Berkeley in 2000 and 1995, and dual BS degrees in Computer Science and Electrical Engineering from Massachusetts Institute of Technology in 1990.

**Brian Harvey – Faculty Advisor**

bh@cs.berkeley.edu  
781 Soda Hall, UC Berkeley

Brian Harvey is a Lecturer SOE in the Computer Science Division of the EECS Department at the University of California, Berkeley. He is the co-author of the current CS3 textbook, *Simply Scheme* as well as the author of *Computer Science Logo Style*, a three book series computer science introduction using the Logo programming language, a predecessor to Scratch. In addition to teaching lower division computer science courses at Berkeley, he currently volunteers at an elementary school teaching students Scratch. Brian received his PhD in education from UC Berkeley in 1985 and Masters in computer science from Stanford University in 1975.

**Jeremy Huddleston – Post-Masters Educational Research Associate**

jeremyhu@cs.berkeley.edu  
535 Soda Hall, UC Berkeley

Jeremy Huddleston's research focus is on interdisciplinary curriculum development for education involving new media. He has designed curricula and taught or co-taught summer courses utilizing Alice or Scratch to middle school students during the summers of 2006, 2007, and 2008 in the PEP, PESA, or SciFY summer programs. In the 2005-2006 academic school year, he designed a year-long course with the support of Pixar, the CS Division, and Art Practice that provides an interdisciplinary approach to teaching students from different backgrounds advanced concepts in digital animation. This course was first offered during the 2006-2007 academic year and is wrapping up its second offering during this 2008-2009 academic year. Jeremy Huddleston expects to receive his MS in computer science education in May 2009 and continue his research and teaching as a post-masters educational research associate.

**Nathaniel Titterton – Research Scientist**

nate@berkeley.edu  
329 Soda Hall, UC Berkeley

Nathaniel works at UC Berkeley as a Research Scientist and Lecturer. He taught the current version of CS3 as a Lecturer at UC Berkeley from 2004-2008, as well as developed and researched the innovative web-based tools used in that and other lower division CS courses. He is currently the project manager for an initiative at UC Berkeley to develop an online curriculum for Advanced Placement computer science. Nathaniel received his PhD in education from UC Berkeley in 2001.

**References**

Astrachan, O., Walker, H., Stephenson, C., Diaz, L., and Cuny, J. 2009. Advanced placement computer science: the future of tracking the first year of instruction. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*

(Chattanooga, TN, USA, March 04 - 07, 2009). SIGCSE '09. ACM, New York, NY, 397-398. DOI= <http://doi.acm.org/10.1145/1508865.1509005>

College Board. (2008). *AP Report to the Nation*. Princeton, NJ: Author. Retrieved March 28, 2009, from <http://professionals.collegeboard.com/data-reports-research/ap/nation/2008>.

diSessa, A. (2000). *Changing Minds: Computers, Learning, and Literacy*. MIT Press:Cambridge, MA.

Farmer, L. (2008). *Teen Girls and Technology: What's the problem, what's the solution?* New York, NY: Teachers College Press.

Garcia, D. D., Cutler, R., Dodds, Z., Roberts, E., and Young, A. 2009. Rediscovering the passion, beauty, joy, and awe: making computing fun again, continued. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA, March 04 - 07, 2009). SIGCSE '09. ACM, New York, NY, 65-66. DOI= <http://doi.acm.org/10.1145/1508865.1508889>

Judson, Eugene. (2008). University of California, Berkeley, Baseline Report for Undergraduate Computer Science and Engineering Programs. National Center for Women & IT.

Malan, D., & Leitner, H. (2007). *Scratch for Budding Computer Scientists. ACM Special Interest Group on Computer Science Education.*, Covington, Kentucky: ACM.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. (2004). Scratch: A sneak preview. *Second International Conference on Creating, Connecting, and Collaborating through Computing*. (pp. 104-109). Kyoto, Japan. Also <http://scratch.mit.edu/>

McGettrick, A., Roberts, E., Garcia, D. D., and Stevenson, C. 2008. Rediscovering the passion, beauty, joy and awe: making computing fun again. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, USA, March 12 - 15, 2008). SIGCSE '08. ACM, New York, NY, 217-218. DOI= <http://doi.acm.org/10.1145/1352135.1352213>

Office of Planning & Analysis, University of California, Berkeley. (2008) CS3-CS61a Fall Update. Berkeley, CA: Author.